Particle Systems

Jungdam Won

Computer Science & Engineering

Seoul National Univ.

This material was created based on the slides and lecture notes of
Physically Based Modeling (SIGGRAPH 2001 course) by Andrew Witkin

Overview

- One lousy particle
- Particle systems
- Forces: gravity, springs, and so on
- Implementation and interaction
- Simple collisions

A Newtonian Particle (Point-mass)

- Differential equation: $\mathbf{f} = m\boldsymbol{a}$
- Forces $\mathbf{f}$ can depend on:
  - Position, Velocity, Time

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$$

Acceleration

Mass

Second Order Equations

- Not in our standard form because it has 2nd derivatives

$$\ddot{\mathbf{x}} = \frac{\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$$

- Add a new variable, $\mathbf{v}$, to get a pair of coupled 1st order euqations

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{f}/m \end{cases}$$

Phase Space (State Space)

- In dynamical systems theory and control theory, a ***phase space*** or ***state space*** is a space in which all possible "states" of a dynamical system or a control system are represented, with **each possible state corresponding to one unique point in the phase space**. - wiki -

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$$

Concatenate x and v to make a 6D-vector:
A position in the phase space

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathrm{v}} \end{bmatrix}$$

Velocity in the phase space: another 6D-vector

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathrm{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}/\mathrm{m} \end{bmatrix}$$

A vanilla 1st-order differential equation

Simulation Pseudocode

Initialize all states

Until $t < T_{max}$

$\mathbf{X}(t) \leftarrow$ Get the current states of all particles

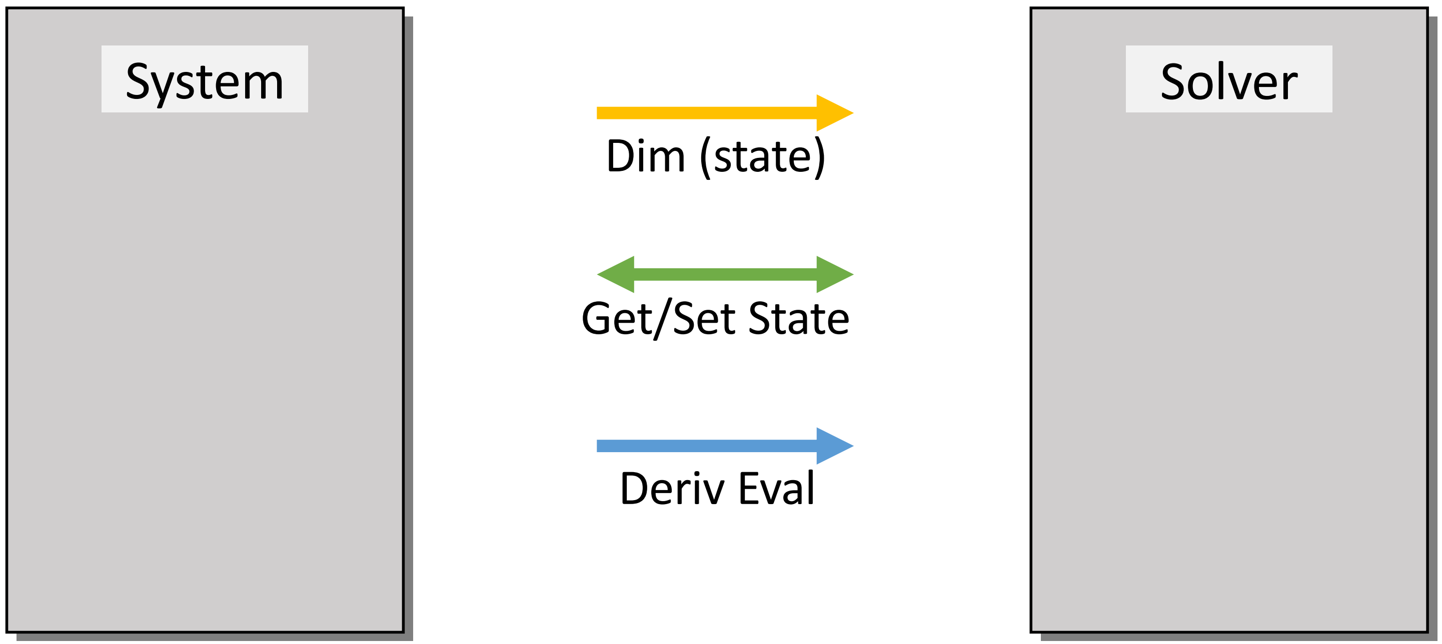
$\mathrm{f} \leftarrow$ Compute forces

$\mathbf{X}(t+h) \leftarrow$ Solve ODEs by using $\mathbf{X}(t), \mathrm{f}$ and other info.

Update states of all particles with $\mathbf{X}(t+h)$

$t \leftarrow t + h$

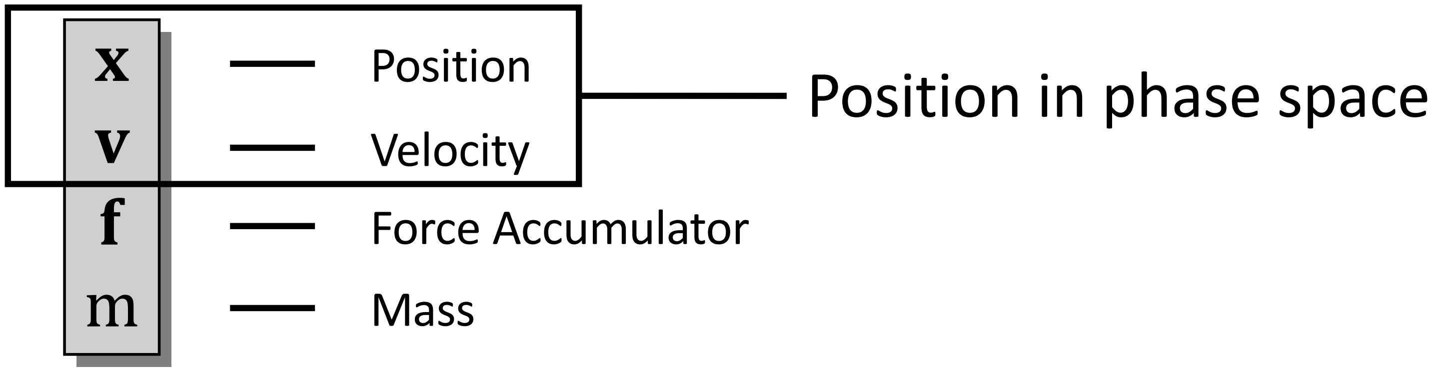
Design of Simulation System



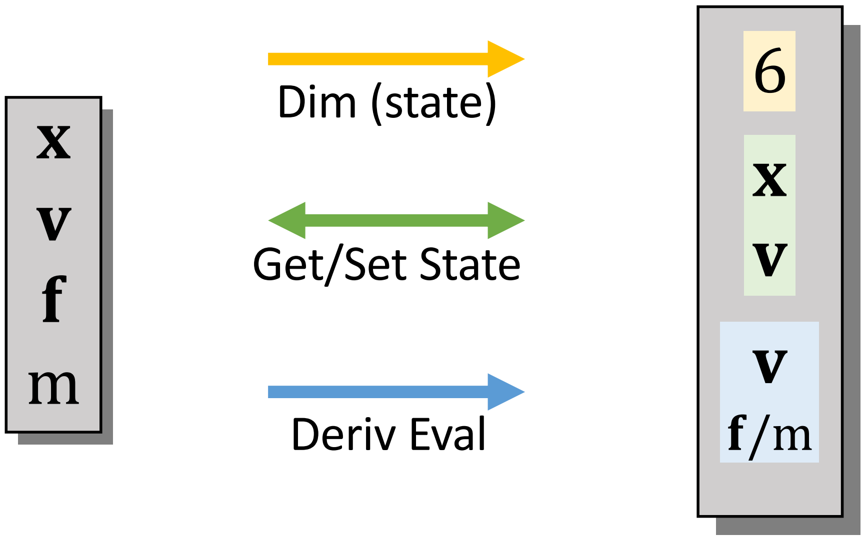
Maintaining the state of our system

Compute forces then compute new state of our system by soloving ODEs

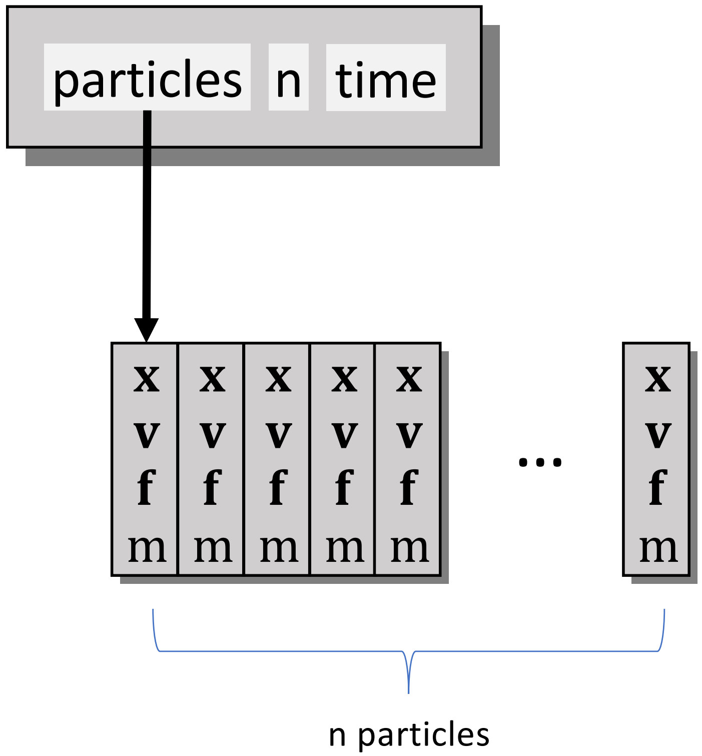
Particle Structure



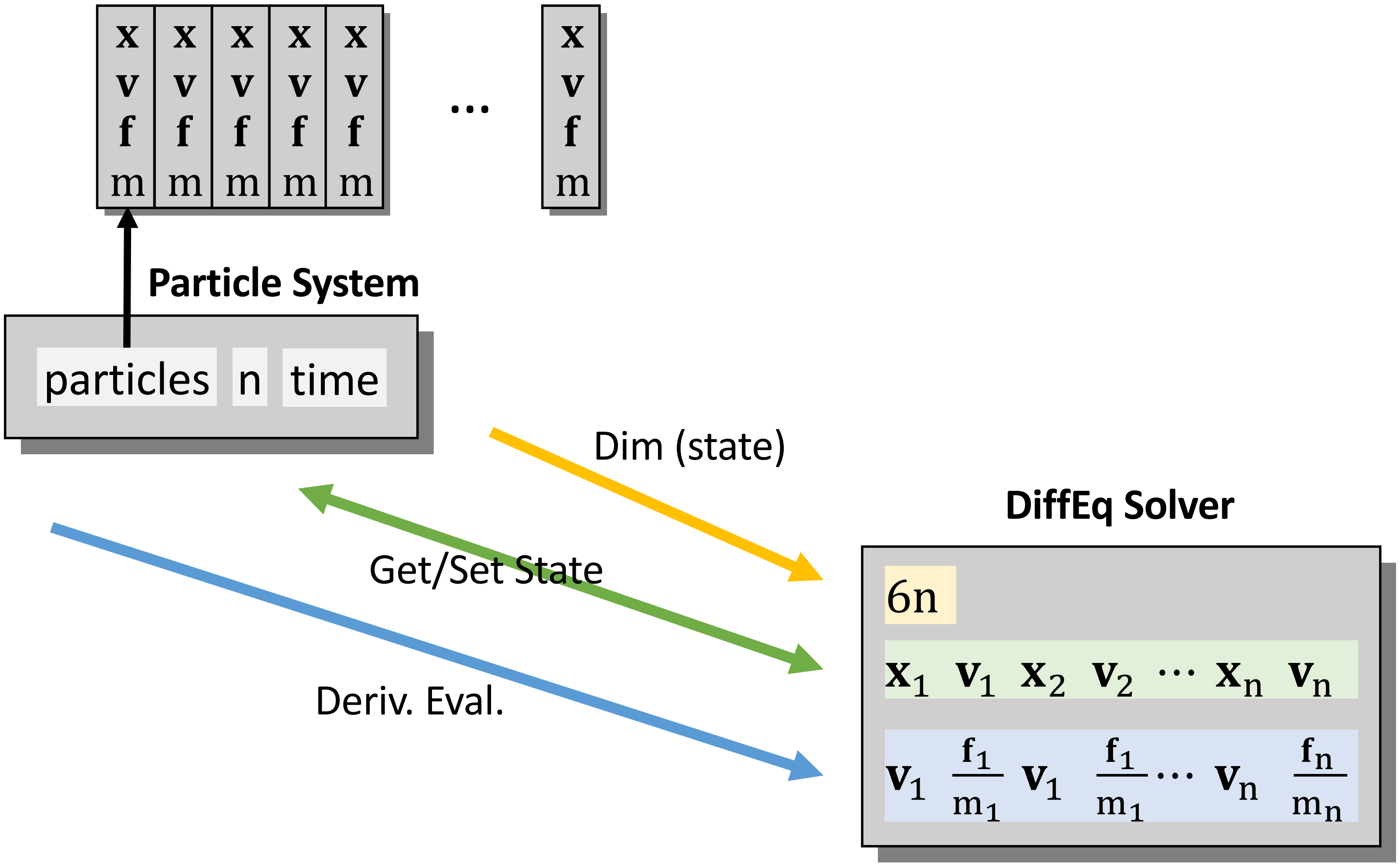
Solver Interface
(for a single particle)



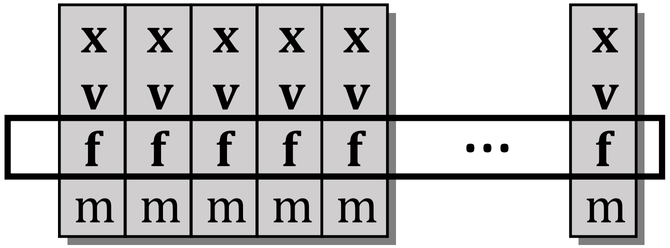
Particle Systems



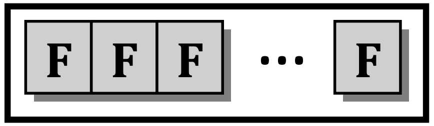
Solver Interface
(for a particle system)



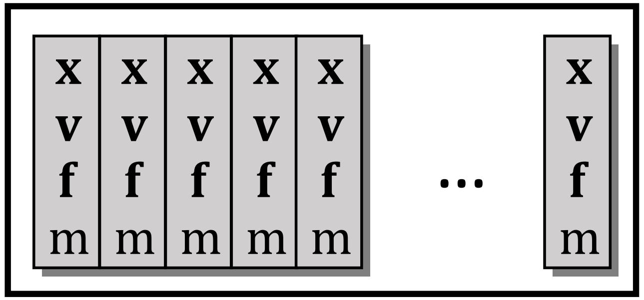
Derivative Evaluation Loop



(1) Loop over particles, zero force accumulators



(2) Caclulate forces by invoking ***apply_force*** functions, sum all forces into accumulators



(3) Return auxiliary values $[\mathbf{v}, \frac{\mathbf{f}}{\mathrm{m}}, \cdots]$ to solver so that it can integrate the state by using them

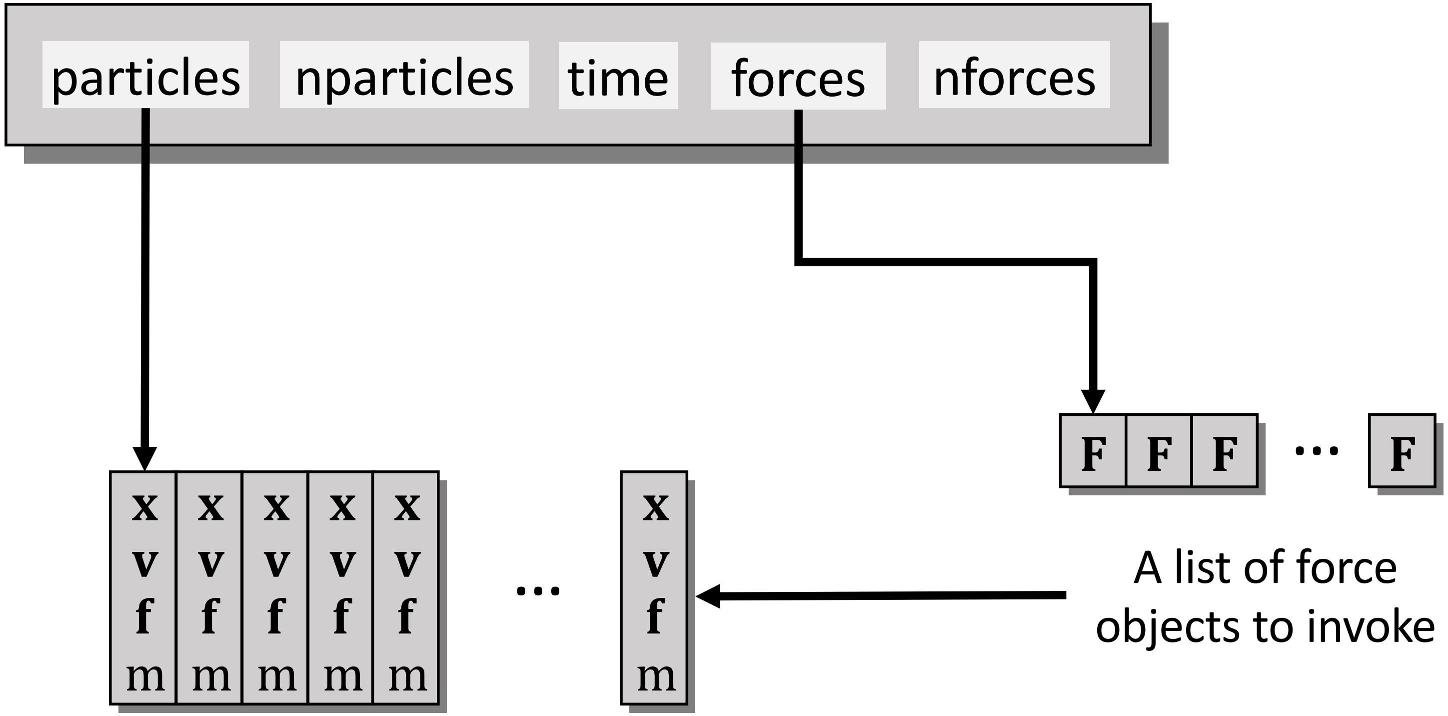
Forces

- Constant
  - Gravity
- Position/time dependent
  - Force fields (e.g. winds)
- Velocity-dependent
  - Drag (i.e. damping)
- n-ary
  - Springs

Force Structures

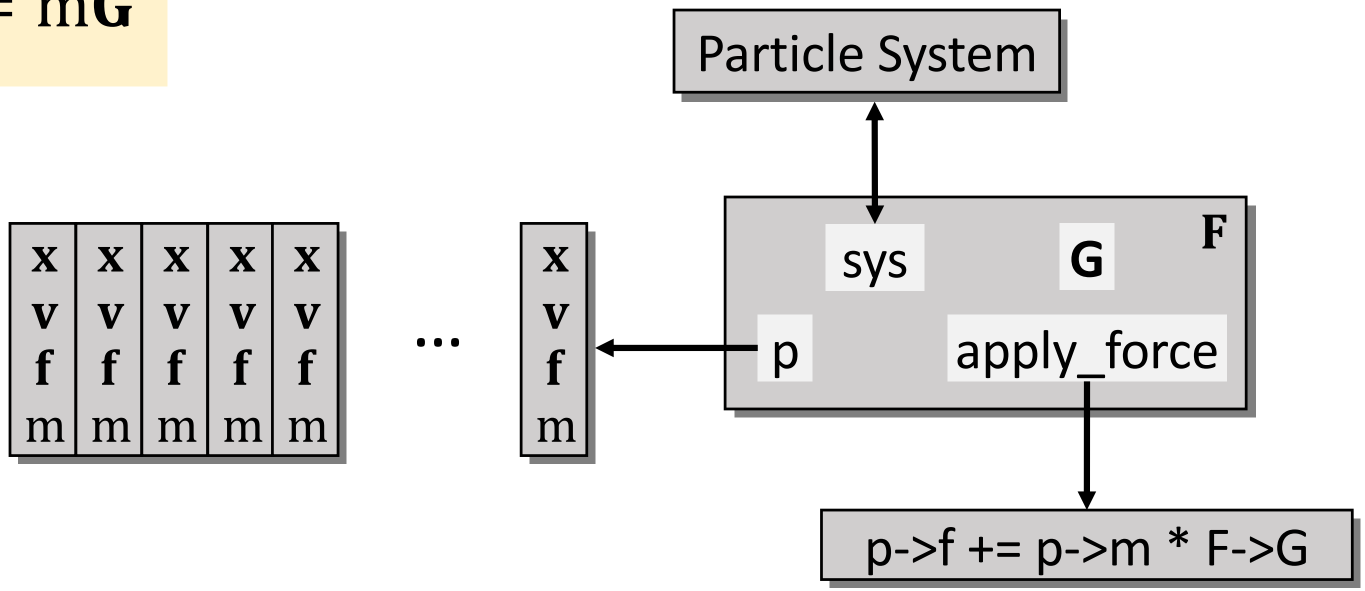
- Unlike particles, forces are ***heterogenous***
- Force objests
  - Will be used as black boxes
  - Pointers to the particles they influence
  - Will have a duty on adding in their own forces (type dependent)
- Global force calculation
  - Loop, invoking force objects

Particle Systems with Forces



Gravity

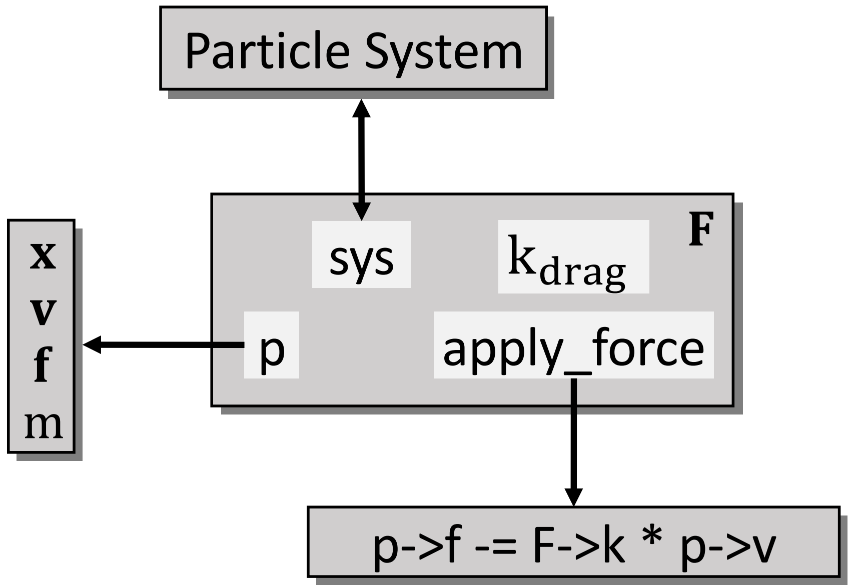
Force Law:
$\mathbf{f}_{\mathrm{grav}} = \mathrm{m}\mathbf{G}$



Viscous Drag

Force Law:

$$\mathbf{f}_{\mathrm{drag}} = -\mathrm{k}_{\mathrm{drag}}\mathbf{v}$$



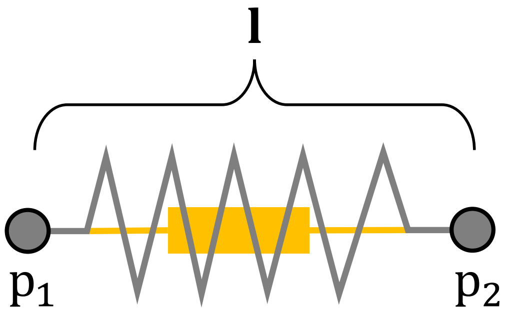
Damped Spring

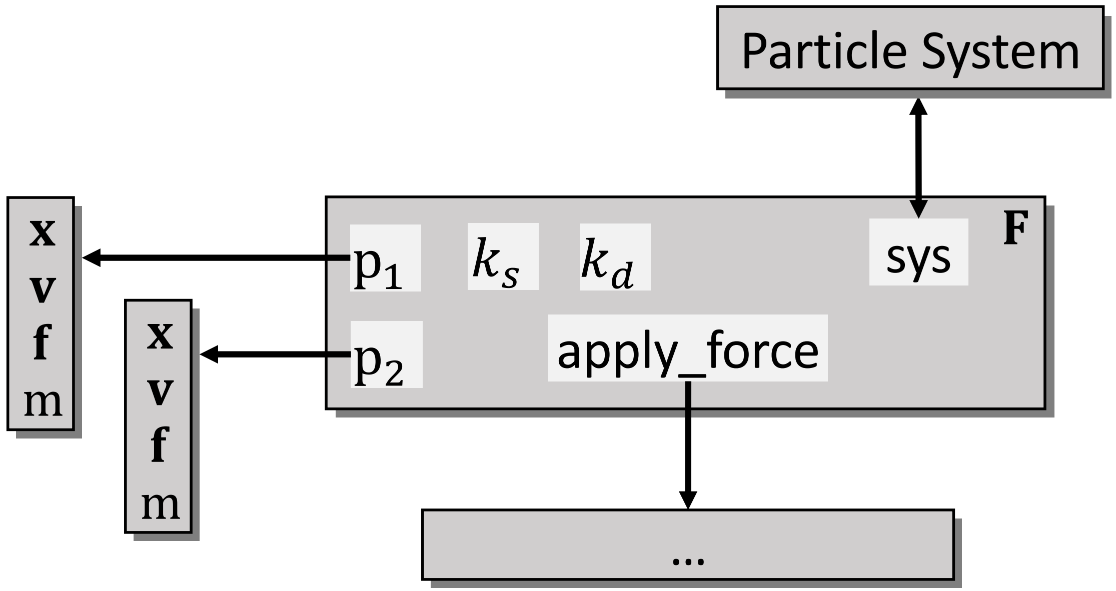
Force Law:

$$\mathbf{f}_1 = -\left[k_s(|\mathbf{l}| - r) + k_d\left(\frac{\dot{\mathbf{l}} \cdot \mathbf{l}}{|\mathbf{l}|}\right)\right]\frac{\mathbf{l}}{|\mathbf{l}|}$$

$$\mathbf{f}_2 = -\mathbf{f}_1$$

where $\mathbf{l} = \mathbf{x}_1 - \mathbf{x}_2$





Energy Functions

- Generically, the position-, velocity-, and time-dependent formulae that we use to calculate forces are known as ***force laws****.* Those laws might not refer to ***real*** laws of physics that are 100% accurate. Rather, they form part of our description of the system we are modeling

- In many cases, it is possible to specify the desired configuration by giving a function that reaches ***zero*** exactly when things are “happy”

- We call this kind of function an ***energy (behavior) function***

Energy Functions: Examples

$$\mathbf{c}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 - \mathbf{x}_2$$

The two particles should be at the same location

$$\mathbf{c}(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2| - r$$

The distance between two particles should be r

$$E = \frac{k_s}{2}\mathbf{c} \cdot \mathbf{c}$$

$$\mathrm{f}_i = \frac{-\partial E}{\partial \mathbf{x}_i} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}_i}(-k_s \mathbf{c})$$

or

$$\mathrm{f}_i = \frac{\partial \mathbf{c}}{\partial \mathbf{x}_i}(-k_s \mathbf{c} - k_d \dot{\mathbf{c}})$$

- ***Constraint-forces*** f_i are computed by taking the gradient of the ***energy functions*** E, where k_s, k_d are stiffness and damping coefficients, respectively
- f_i can be regarded as a generalized spring forces that attract the system to states that satisfy $\mathbf{c} = 0$

Energy Functions: Examples

$$\mathrm{f}_i = \frac{\partial \mathbf{c}}{\partial \mathbf{x}_i}(-k_s\mathbf{c} - k_d\dot{\mathbf{c}})$$

$$\mathbf{c}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 - \mathbf{x}_2 = \begin{bmatrix} x_{11} - x_{21} \\ x_{12} - x_{22} \\ \vdots \\ x_{1\mathrm{n}} - x_{2\mathrm{n}} \end{bmatrix} \in R^{\mathrm{n}}$$

$$\dot{\mathbf{c}} = \mathbf{v}_1 - \mathbf{v}_2, \qquad \frac{\partial \mathbf{c}}{\partial \mathbf{x}_1} = \mathrm{I}, \qquad \frac{\partial \mathbf{c}}{\partial \mathbf{x}_2} = -\mathrm{I}$$

$$\mathbf{f}_1 = -k_s(\mathbf{x}_1 - \mathbf{x}_2) - k_d(\mathbf{v}_1 - \mathbf{v}_2)$$

$$\mathbf{f}_2 = k_s(\mathbf{x}_1 - \mathbf{x}_2) + k_d(\mathbf{v}_1 - \mathbf{v}_2)$$

Energy Functions: Examples

$$\mathrm{f}_i = \frac{\partial \mathbf{c}}{\partial \mathbf{x}_i}(-k_s\mathbf{c} - k_d\dot{\mathbf{c}})$$

$$\mathbf{c}(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{l}| - r \in R \quad \text{where } \mathbf{l} = \mathbf{x}_1 - \mathbf{x}_2$$

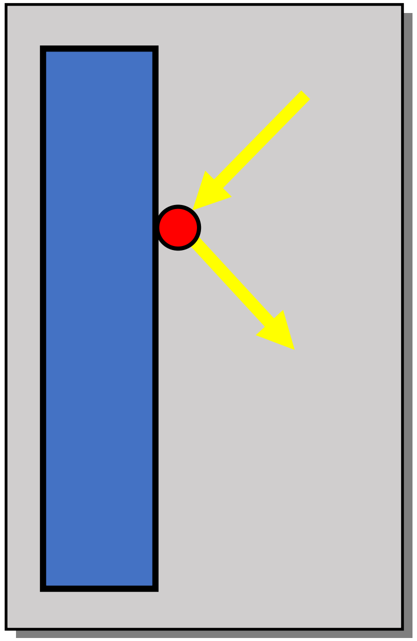
$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}_1} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}_1} = \frac{\mathbf{l}}{|\mathbf{l}|}, \qquad \frac{\partial \mathbf{c}}{\partial \mathbf{x}_2} = -\frac{\mathbf{l}}{|\mathbf{l}|}$$

$$\frac{\partial \mathbf{c}}{\partial \mathbf{l}} = \frac{\mathbf{l}}{|\mathbf{l}|} \qquad \dot{\mathbf{c}} = \frac{\partial \mathbf{c}}{\partial \mathbf{l}}\frac{\partial \mathbf{l}}{\partial \mathrm{t}} = \dot{\mathbf{l}} \cdot \frac{\mathbf{l}}{|\mathbf{l}|}$$

$$\mathbf{f}_1 = -\left[k_s(|\mathbf{l}| - r) + k_d(\dot{\mathbf{l}} \cdot \frac{\mathbf{l}}{|\mathbf{l}|})\right]\frac{\mathbf{l}}{|\mathbf{l}|}$$

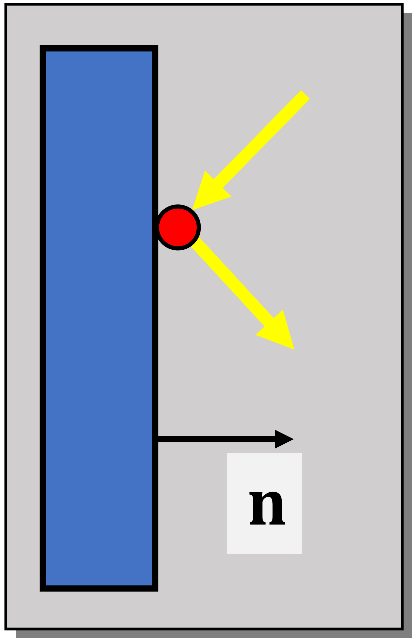
$$\mathbf{f}_1 = \left[k_s(|\mathbf{l}| - r) + k_d(\dot{\mathbf{l}} \cdot \frac{\mathbf{l}}{|\mathbf{l}|})\right]\frac{\mathbf{l}}{|\mathbf{l}|}$$

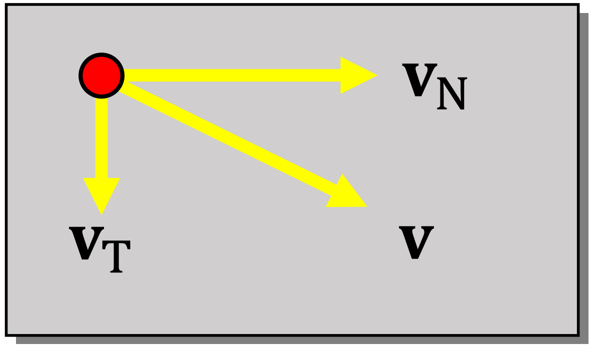
Bouncing off the Walls



- Later: rigid body collision and contact
- For now, just simple point-plane collisions
- Add-ons for a particle simulator

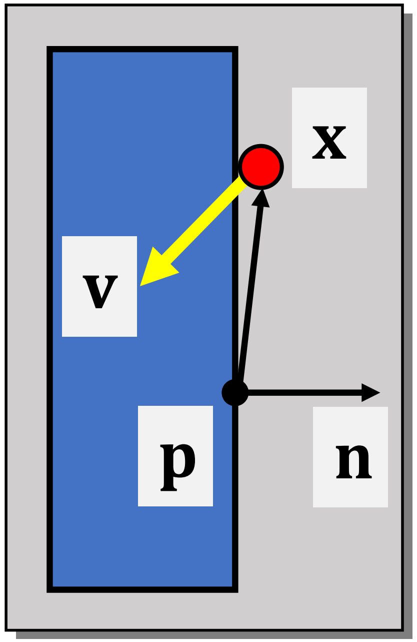
Normal and Tangential Components





$$\mathbf{v}_\mathrm{N} = (\mathbf{n} \cdot \mathbf{v})\mathbf{n}$$
$$\mathbf{v}_\mathrm{T} = \mathbf{v} - \mathbf{v}_\mathrm{N}$$

Colliding Contact



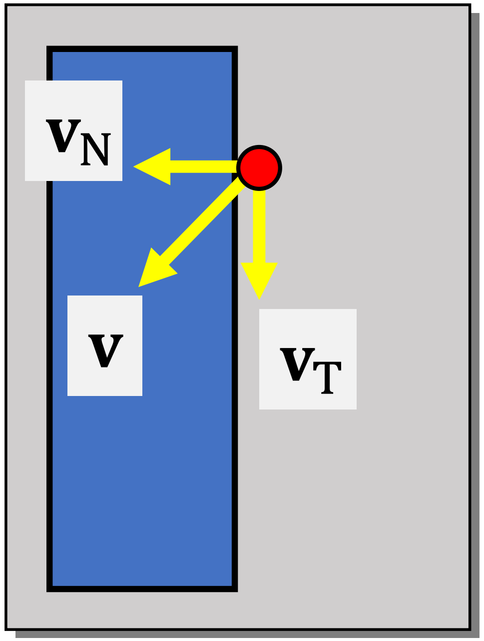
$$(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} < \epsilon$$

"The point is very close to the wall"

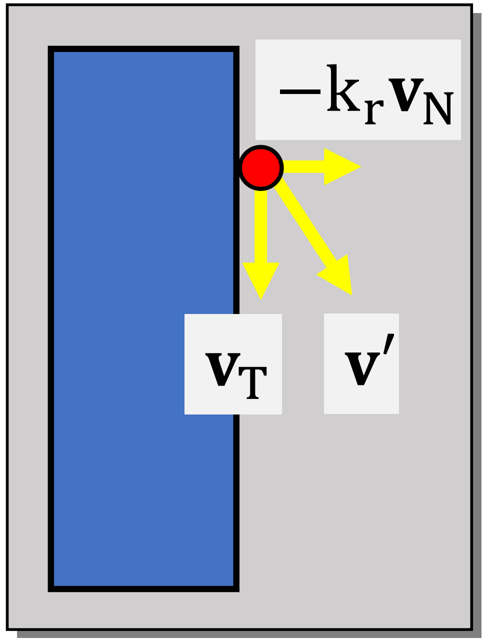
$$\mathbf{n} \cdot \mathbf{v} < 0$$

"The point is moving toward to the wall"

Colliding Contact



Before collision

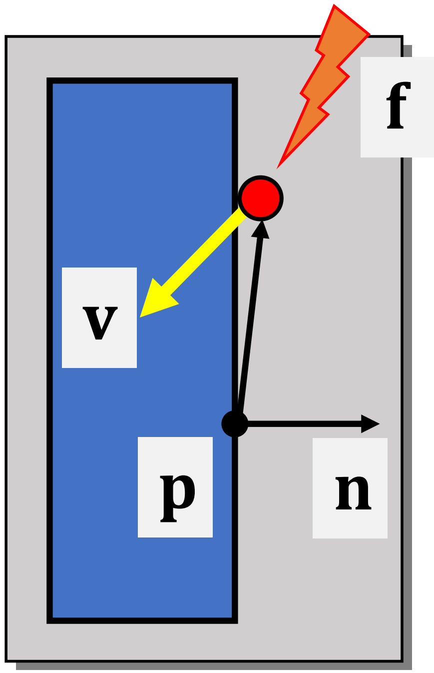


After collision

$$\mathbf{v}' = \mathbf{v}_\mathrm{T} - \mathrm{k_r}\mathbf{v}_\mathrm{N}$$

$\mathrm{k_r}$: coefficient of restitution

Resting Contact



If the particle is on the collision surface, and

$$|(\mathbf{x}-\mathbf{p})\cdot\mathbf{n}| < \epsilon$$

$$|\mathbf{n}\cdot\mathbf{v}| < \epsilon$$

If the particle is pushed into the wall ($\mathbf{n}\cdot\mathbf{f} < \mathbf{0}$)

$$\mathbf{f}_{c} = (-\mathbf{n}\cdot\mathbf{f})\mathbf{n}$$

"The wall pushed back"

$$\mathbf{f}_{f} = -k_f(-\mathbf{n}\cdot\mathbf{f})\mathbf{v}_{\mathbf{T}}$$

"The friction forces are generated due to the normal force"

k_f: fricitional coefficient

This is a simple linear friction model!

Basic User Interaction